Name: $\boxed{\text{Sahib Bajwa}}$
ID: $\boxed{107553096}$

CSCI 3104, Algorithms **Profs. Chen & Grochow**
Problem Set 8 – Due Thurs Apr 2 11:55pm **Spring 2020, CU-Boulder**

Advice 1: For every problem in this class, you must justify your answer: show how you arrived at it and why it is correct. If there are assumptions you need to make along the way, state those clearly.

Advice 2: Informal reasoning is typically insufficient for full credit. Instead, write a logical argument, in the style of a mathematical proof.

Instructions for submitting your solutions:

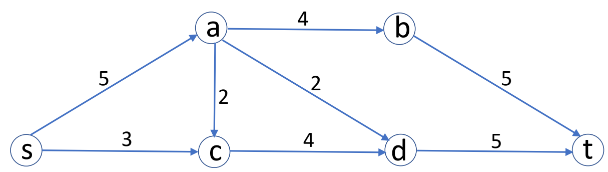
- All submissions must be typed.
- You should submit your work through the **class Canvas page** only.
- You may not need a full page for your solutions; pagebreaks are there to help Gradescope automatically find where each problem is. Even if you do not attempt every problem, please allot at least as many pages per problem (or subproblem) as are allotted in this template.

Quicklinks: 1a 1b 2a 2b

Name: $\boxed{\text{Sahib Bajwa}}$
ID: $\boxed{107553096}$

CSCI 3104, Algorithms **Profs. Chen & Grochow**
Problem Set 8 – Due Thurs Apr 2 11:55pm **Spring 2020, CU-Boulder**

1. *Consider the following flow network, with each edge labeled by its capacity:*

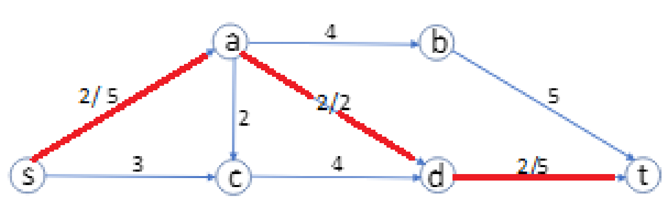


(a) Using the Ford-Fulkerson algorithm, compute the maximum flow that can be pushed from $s \to t$. **You must use** $s \to a \to d \to t$ **as your first flow-augmenting path.**

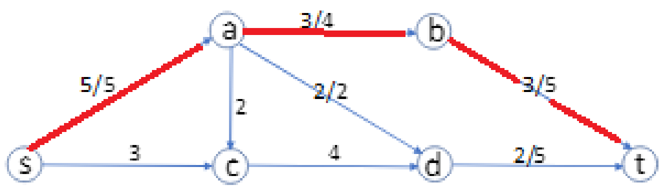
In order to be eligible for full credit, you must include the following:

- The residual network for each iteration, including the residual capacity of each edge.
- The flow augmenting path for each iteration, including the amount of flow that is pushed through this path from $s \to t$.
- The updated flow network **after each iteration**, with flows for each directed edge clearly labeled.
- The maximum flow being pushed from $s \to t$ after the termination of the Ford-Fulkerson algorithm.

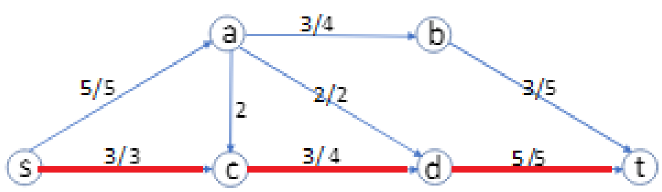
For the first flow argument path, we use $s \to a \to d \to t$. The residual capacity of $s \to a$ will be 2/5. The residual capacity of $a \to d$ will be 2/2. The residual capacity of $d \to t$ will be 2/5. The rest of the edges will be unchanged and currently have 0 flow being pushed through them. The amount of flow that is pushed through this path from $s \to t$ is 2. Here is the updated flow network after this iteration:



For the second flow argument path, we use $s \to a \to b \to t$. The residual capacity of $s \to a$ will be 5/5. The residual capacity of $a \to b$ will be 3/4. The residual capacity of $b \to t$ will be 3/5. The rest of the edges are unchanged from the previous iteration. The amount of flow that is pushed through this path from $s \to t$ is 3. Here is the updated flow network after this iteration:



For the third flow argument path, we use $s \to c \to d \to t$. The residual capacity of $s \to c$ will be 3/3. The residual capacity of $c \to d$ will be 3/4. The residual capacity of $d \to t$ will be 5/5. The rest of the edges are unchanged from the previous iteration. The amount of flow that is pushed through this path from $s \to t$ is 3. Here is the updated flow network after this iteration:



considering both of the edges leading out of s are full and no flow is lost from those two edges to t, the Ford-Fulkerson algorithm should terminate. The maximum flow being pushed from $s \to t$ after the termination of the Ford-Fulkerson algorithm will be 8.

(b) The Ford-Fulkerson algorithm will terminate when there is no longer an augmenting path on the residual network. At this point, you can find a minimum capacity cut. Indicate this cut and its capacity, and verify if max flow min cut theorem holds.

One example for the minimum cut is (s, a), (s, c). (I believe that (b, t), (d, t) also works as a minimum cut)
This cut has a capacity of 8. The max flow min cut theorem says that the minimum cut capacity should be equal to the maximum flow of the algorithm. Since the minimum capacity cut and the maximum flow of the graph are both 8, the max flow min cut theorem holds.

2. In this problem, we seek to generalize the max-flow problem from class to allow for multiple sources and sinks. Given a directed graph $G = (V, E)$ with capacity $c(u, v) > 0$ for each edge $(u, v) \in E$ and demand $r(v)$ at each vertex $v \in V$, a routing of flow is a function f such that

   - **(capacity contraint)** for all $(u, v) \in E$, $0 \leq f(u, v) \leq c(u, v)$, and
   - **(flow conservation)** for all $v \in V$,

   $$\sum_{u:(u,v)\in E} f(u, v) - \sum_{u:(v,u)\in E} f(v, u) = r(v),$$

     i.e., the total incoming flow minus the total outgoing flow at vertex v is equal to $r(v)$.

   We note that $r(v)$ can be positive, negative, or 0, just as in the max-flow setting from class. In particular, note that:

   - The vertex v is a **source** vertex precisely if $r(v) < 0$.
   - The vertex v is a **sink** vertex precisely if $r(v) > 0$.
   - The vertex v is neither a source nor a sink precisely if $r(v) = 0$.

   These conditions are the same as in the single-source, single-sink max-flow model from class.

   (a) Show how to find a routing or determine that one does not exist by reducing to a maximum flow problem. In other words, given a graph, c and r as above, construct a related graph H with capacities $c' \colon E(H) \to \mathbb{R}$ and $s, t \in V(H)$ such that you can use the maximum $s \to t$ flow on H to determine whether a routing exists in G. Hint: You may want to take a look at Chapter 7.5 of the recommended text by Kleinberg and Tardos to get some insight.

To determine a routing for the graph, we need to change the problem into a single-source, single-sink max-flow model. To do this, we create a source node s and a sink node t. Considering we know the r(v) values of every vertex, we can use them to create our new graph (I would imagine this graph to look similar to figure 7.11 in Chapter 7.5 of the mentioned textbook). If the r(v) value of a vertex is ¡ 0, we know the vertex is a source. If the vertex is a source, we will connect it to the source node s we created earlier. If the r(v) value of a vertex is ¿ 0, then it is a sink. If the vertex is a sink, we will connect it to the sink node t we created earlier. If the r(v) value of a vertex is = 0, we will leave it alone as it has as much flow going in as flow going out. We do this for all vertexes in the graph until we end up with a single-source, single-sink max-flow model. Since we have this model now, we can run the Ford-Fulkerson algorithm until it produces a maximum flow on the graph. Since (or if) we have a maximum flow from $s \to t$, we have a routing that exists in G.

(b) Suppose that additionally you are given a lower bound $l(u, v) > 0$ at each edge (u, v), and we are looking for a routing f satisfying

- **(lower bounds)** for all $(u, v) \in E$, $f(u, v) \geq l(u, v)$,

in addition to the **capacity constraint** and the **flow conservation**. Show how to find such a routing or determine that one does not exist by reducing to a maximum flow problem. Hint: You may want to solve this problem by reducing it to (a). Think about how to modify the graph and demands to equivalently enforce lower bounds $l(u, v)$ on the flows.

If we go back to the graph constructed in (a), we can use it to solve this problem. Before each edge with a lower bound, we can add a new vertex that connects to the starting vertex of that edge. We can add a vertex like this to every source vertex for each edge. Using our new vertexes, we can change the capacity of the new edge we have created to direct the flow from the source vertex we created in (a) in order to provide each edge with however much flow it needs (f(u,v) $\geq$ l(u,v)). We can then run the Ford-Fulkerson algorithm to determine the maximum flow for the graph. Since we are able to manipulate the amount of flow going to each edge, we can also manipulate the maximum flow provided to us by the Ford-Fulkerson algorithm. If we get a maximum flow that satisfies f(u,v) $\geq$ l(u,v) for each edge (u,v), then such a routing does exit. If we cannot, then such a routing does not exist.